git tutorial

all you need to know about git & github aka 瞎聊git

linxinhui, AMaze

下载和安装git

在此之前，什么是git?
当然还有github

- 当然可以直接从这里下载软件包并安装，各种操作系统都一样
- 但是我们第一次讨论的时候花了两个小时解释什么是包管理器以及为什么应该使用包管理器。于是安装git的过程就变成下面这样：
    - 在Linux下，具体哪一种包管理工具取决于发行版，我们以Ubuntu为例：`$ sudo apt install git`
    - MacOS用户的操作可以用brew：`$ brew install git`，当然你直接在命令行中输入git（在没有安装的情况下）会触发Xcode command line toolkit（之类的名字）执行git安装过程
    - Windows也有包管理工具，比较成熟的或许是scoop和chocolatey，微软自己也在开发包管理工具，叫winget，对于Windows包管理工具，本人长期心向往之但懒得折腾，大家可以自己尝试（要是和我一样懒得折腾，直接下载exe文件或者用WSL），这里提供了一个安装过程录制视频
- 据不严谨观察，很多人（尤其是非计算机专业的同学）的认知过程是先github再git（或许没有git），这里简单介绍两者的关系：git是版本管理工具，github是git的衍生品，是免费的远程仓库，来自全世界的开发者将自己的工程公开发表在github上，使得参与别人的项目和让别人参与自己的项目变得非常容易，github是全球最大的~~同性交友网站~~开源协作社区

配置git

the very beginning

- 通过查看版本验证某个软件/包是否安装完成是一种简单而有效的手段（当然随着安装的东西越来越复杂这种手段的有效性越来越弱）：`git -v`
- Git是一个分布式管理系统，这意味着在使用git前需要自报家门，$(意味着的内容)意味着在安装好git后需要先“自报家门”：

```
$ git config –global user.name <user_name>

$ git config –global user.email <user_email>
```

创建git仓库

Initialized empty Git repository in …

- 开始之前，我们或许先回顾几个基本的操作文件和目录操作命令，比如cd、ls（包括ls -al）、mkdir、touch和rm（包括rm -rf）
- （还需要介绍什么是git仓库么）然后介绍如何创建git仓库：
    - 将某个目录设置为git仓库，切换到该目录下，用$ git init命令创建git仓库，此时对应目录下出现一个.git（隐藏）目录
    - $ git status命令可以查看当git仓库的信息，包括一个目录是不是git仓库
    - 不再将某个目录作为git仓库，删除对应.git即可
    - 简单演示一下
- 在github上创建仓库的方法简单很多，基本不涉及命令行，需要注意的是（这里有一个小tip之后介绍本地和远程仓库同步的时候再说）

通过git进行版本管理

version control is all you need

- 版本管理是git最重要的功能
- 创建git仓库（版本库）之后，如何把文件添加到版本库/如何跟踪一个文件的修改?
    - 首先介绍工作区（work directory）和暂存区（stage）两个基本概念，工作区是工作的地方（废话），在这个例子中git_tutorial目录（不包含.git目录）就是工作区，暂存区是暂时存放工作区内容的目录（还是废话），暂存区在.git目录下（.git目录下保存暂存区、分支和指向某一分支的某一提交的指针），文件在从工作区被提交（commit）到版本库（的某一个分支，不用管这个括号里的东西）之前需要先被添加（add）到暂存区
    - `$ git add <file_name>`将文件添加到暂存区
    - `$ git commit -m <commit_message>`提交暂存区中的文件到git仓库（的某个分支上）
    - `$ git log`查看提交记录，用`$ git log –pretty=oneline`查看更短的提交记录
    - 不想将某些东西添加到git仓库中（比如编译出来的二进制文件，或者MacOS下的.DS_Store文件等），新建一个.gitignore文件，在里面写上不准备添加到git仓库中的文件的路径即可
    - 某个文件被修改了，想知道具体修改了什么内容? 用`$ git diff <file_name>`查看
    - 演示一下

通过git进行版本管理

version control is all you need x2

- 版本管理是git最重要的功能 x2
- 提交了若干个版本后，如何回退到某个历史版本?
    - 每次commit之后git都会生成一个commit id，通过`$ git log`查看某次提交的id后，用`$ git reset –hard <commit_id>`命令进行版本回退
    - 后悔药？回退到某个历史版本后看看提交记录？如何回到未来？通过`$ git relog`可以查看“log的log”，通过relog中查询的“未来的commit_id”可以回到“未来的版本”
    - git的版本切换和HEAD指针密切相关，reset的过程本质是切换HEAD指针位置的过程
    - 演示一下
- git管理的是修改而不是每次修改后的文件，以上介绍的内容是当修改已经通过工作区编辑、暂存区暂存最终提交到git版本库后可以进行的操作，当修改只发生在工作区，或仍在暂存区（没有提交到分支）时git也能管理修改（主要是撤销），这部分内容不赘述，有兴趣可以自行阅读文档

同步远程仓库和本地仓库

local git repository & remote git repository

- 可能是最重要的命令，通过`$ git clone <github_repo_url>`将github上的项目克隆到本地（别再下载压缩包了，也别用什么Github Desktop了，球球！）
- （填坑的时候到了）如果分别创建本地仓库和远程仓库，建立连接的过程比较麻烦，有时候我们只在github上创建一个空项目，将仓库克隆到本地再进行后续操作
- 开始之前，请先注册github账号
- 本地仓库和远程仓库之间传输通过ssh加密（或许你想了解一下什么是ssh），简单地说，进行传输之前，先配置密钥（公钥和私钥）：
    - ssh key一般存在用户家目录下的.ssh目录下，通常命名为id_rsa和id_rsa.pub
    - 如果没有这两个文件，手动生成一下：`$ ssh-keygen -t rsa -C <your_email_address>`
    - 请妥善保管id_rsa（私钥），将id_rsa.pub（公钥）的内容传递给github
    - 除了ssh公钥外，还需要配置token（或许你想了解一下什么是token），在github设置中选择开发者设置，开放必要权限，生成token，复制生成的token，在终端中操作要求输入github用户名和账号时将此token用作密码输入即可
    - 循环“写代码、提交代码、修改代码、提交代码”一系列流程，直到准备将代码同步到github仓库，假设此时本地仓库处于working tree clean状态且本地仓库和远程仓库没有任何冲突，使用`$ git push`命令同步
    - 演示一下

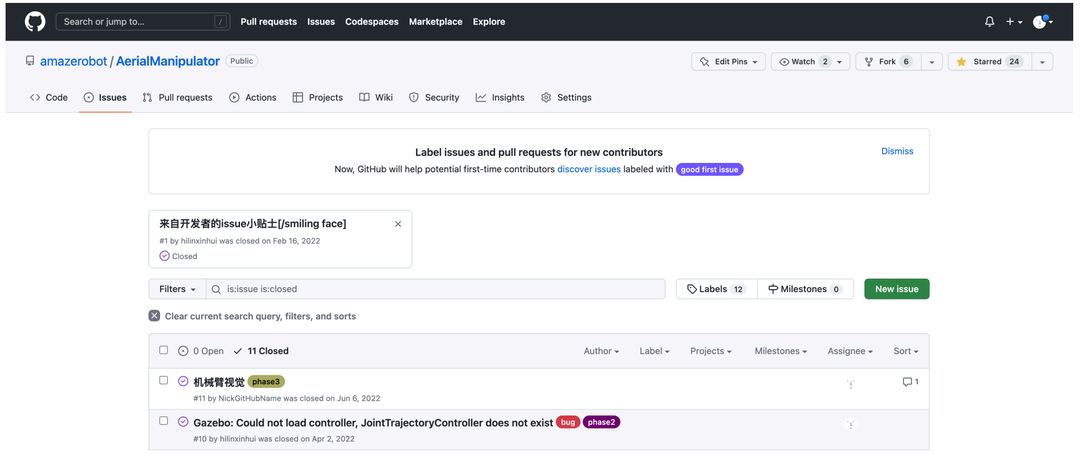
其他需要知道的关于git的东西

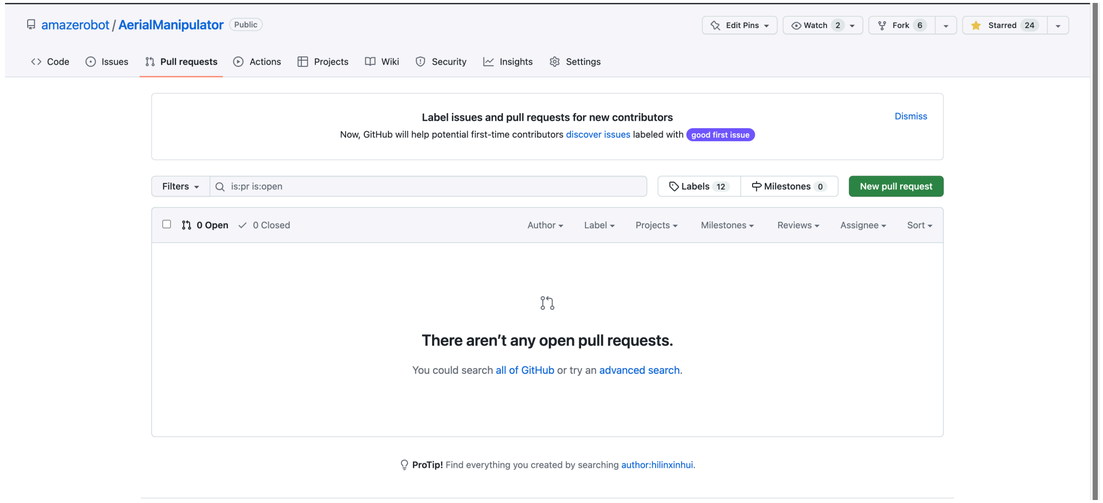
branch & tag

- branch就像平行宇宙
    - 大家已经知道了环境隔离的必要性，在多人合作的项目中，我们显然也希望分隔出自己的工作区
    - 以飞行机械臂为例，大家新建各自的分支完成不同的任务，完成开发后由我合并到主分支（main/master）上
    - 对简单的项目而言，分支管理难度不大，大家在自己的分支上工作，不需要关心其他分支的状态，大家完成工作后，各自的分支可能产生的冲突也几乎没有，对合并非常友好（毕竟分支管理最困难的部分就是处理合并时的冲突）
- tag机制提供了一种映射
    - 类似DNS实现了从复杂ip地址到简单域名的映射，tag将复杂的commid id映射为简单的version id
    - 飞行机械臂项目就使用tag划分环境配置、机械臂建模和运动控制、机械臂视觉等模块，这个视频介绍了具体操作
    - tag和release有莫大的联系，但这点我们以后再介绍

进一步了解github

Github Oriented Programming，拿来主义和站在巨人的肩膀上





阅读资料

多看点总没坏处

- Pro Git，官方唯一指定教材，当做工具书检索即可
- 廖雪峰老师的git教程对得起深入浅出的评价，也是我的启蒙教程
- 2022年2月录的视频包含今天讲的大部分内容，演示可能更清楚一点
- MIT的missing semester系列课程也是众多神作之一，大家遇上了一个好时代，这门课的所有内容基本都有人翻译了，这门课的第六节给出了一个非常基本的git实现，有兴趣可以看看
- 欢迎补充~